****



**Wols 1.80**

Antonio De Padova, Antonio Bottalico e Domenico de Dato

**Sommario**

[**Contesto e obiettivi**](#_heading=h.30j0zll) **[2](#_heading=h.30j0zll)**

[**Analisi dello stato dell’arte**](#_heading=h.1fob9te) **[2](#_heading=h.1fob9te)**

[Analisi caratteristiche per prezzo robot aspirapolvere](#_heading=h.3znysh7) [3](#_heading=h.3znysh7)

[Hardware open source](#_heading=h.tyjcwt) [4](#_heading=h.tyjcwt)

[Software open source](#_heading=h.2s8eyo1) [6](#_heading=h.2s8eyo1)

[**Disegno realizzazione e test**](#_heading=h.17dp8vu) **[7](#_heading=h.17dp8vu)**

[Specifica dei requisiti](#_heading=h.3rdcrjn) [7](#_heading=h.3rdcrjn)

[Progetto fisico](#_heading=h.jdlbcesrftip) [7](#_heading=h.jdlbcesrftip)

[Modello](#_heading=h.fdezkeziu22) [8](#_heading=h.fdezkeziu22)

[Schema del software](#_heading=h.2jxsxqh) [9](#_heading=h.2jxsxqh)

[Implementazioni](#_heading=h.z337ya) [11](#_heading=h.z337ya)

[Test](#_heading=h.3j2qqm3) [26](#_heading=h.3j2qqm3)

[**Conclusione e sviluppi futuri**](#_heading=h.1y810tw) **[26](#_heading=h.1y810tw)**

[**Matrice RACI**](#_heading=h.4i7ojhp) **[27](#_heading=h.4i7ojhp)**

[**Bibliografia e sitografia**](#_heading=h.2xcytpi) **[28](#_heading=h.2xcytpi)**

[**Appendici**](#_heading=h.147n2zr) **29**

# 

# **Contesto e obiettivi**

Il seguente è un progetto a cura di Bottalico Antonio, De Padova Antonio e de Dato Domenico. Siamo tre studenti iscritti al primo anno della facoltà di “Informatica e Tecnologie per la Produzione del Software” presso l’Università degli studi di Bari Aldo Moro che, grazie al professor Mario A. Bochicchio, si sono posti come scopo quello di progettare e sviluppare un robot aspirapolvere.

# **Analisi dello stato dell’arte**

Sempre più abitazioni sono dotate di un aspirapolvere robot, infatti, chi lo usa lo vede essenziale come un qualsiasi altro elettrodomestico. Questa piccola apparecchiatura si muove in totale autonomia in tutta la casa svolgendo le funzioni di spazzare e aspirare, il che è molto conveniente, visto il tempo recuperato che è invece possibile dedicare a una qualsiasi altra attività. Nel 1996 James Dizon[[1]](#bookmark=id.1ci93xb) inventa il primo prototipo di robot aspirapolvere, in grado di funzionare autonomamente, senza filo e senza l’intervento umano. L’invenzione di Dizon venne subito acquistata dalla Electrolux, azienda produttrice di elettrodomestici, che in breve tempo immette sul mercato il primo vero robot aspirapolvere: il trilobite. Il secondo tentativo arriva con il robot aspirapolvere Dyson DC06, ma anche questo modello, a causa di un prezzo altissimo, non riesce a conquistare i consumatori; l’invenzione del robot aspirapolvere rimane, come spesso succede nella storia delle nuove tecnologie, in una sorta di “limbo”. A distanza di diversi anni, nel 2002, con l’introduzione del primo Roomba, da parte della statunitense iRobot, i robot aspirapolvere diventano un bene di consumo reale. Dopo un test iniziale, con la produzione di sole, precisamente 15.000 unità, presto Roomba vedrà conferma del proprio successo e il suo robot aspirapolvere spopolerà, portando all’azienda enormi guadagni. Di fatto il termine “Roomba” è diventato quasi sinonimo di robot aspirapolvere. Dai primi robot aspirapolvere ad oggi, la situazione si è notevolmente evoluta, e continua a farlo, con un affinamento delle capacità di pulizia, un’intelligenza sempre più autonoma, la capacità di mappare strutture, sensori avanzatissimi e possibilità di controllo remoto.

## Analisi caratteristiche per prezzo robot aspirapolvere

(fonte: Amazon)

| **Prezzo** | **Sotto i 100 euro** | **Media fascia** | **Sopra i 300 euro** |
| --- | --- | --- | --- |
| Durata batteria | <100 minuti | <150 minuti | >150 minuti |
| Controllo con app e mappatura | No | Si | Si |
| Controllo con telecomando | Si | Si | Si |
| Lavapavimenti | Alcuni | Alcuni | Si |
| Ricarica automatica | No | Si | Si |
| Svuotamento automatico | No | No | Si |
| Compatibile con Alexa | No | Si | Si |
| Compatibile con Google Home | No | Si | Si |
| **Note:** La rumorosità, l’altezza, il peso, la potenza di aspirazione e l’intelligenza sono direttamente proporzionali alle fasce di prezzo nella maggior parte dei prodotti. | | | |

Alla luce delle caratteristiche elencate precedentemente la fascia in cui vorremmo competere è quella più economica, puntando ad offrire al pubblico elementi come la mappatura e il controllo tramite l’app che su questa fascia mancano.

## 

## Hardware open source

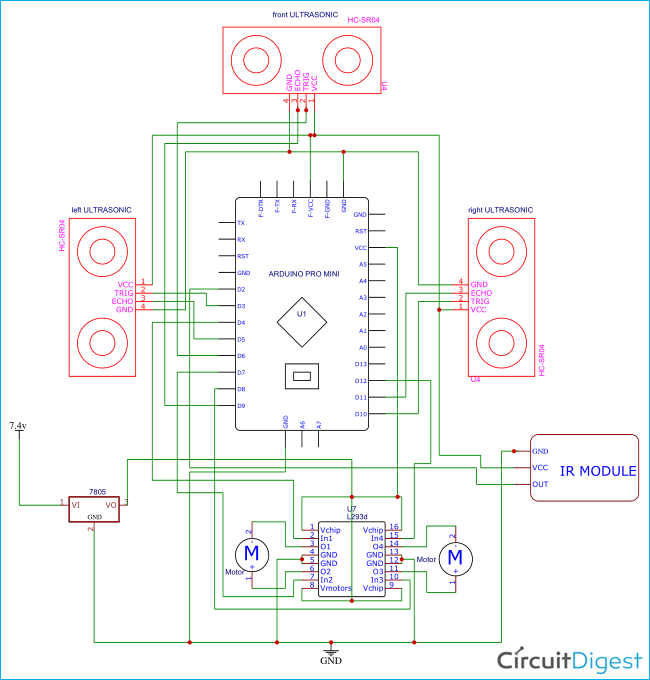
Il robot aspirapolvere Arduino costruito da Mr.X[[2]](#bookmark=id.3whwml4) è molto compatto e pratico, possiede sensori ad ultrasuoni e un sensore di prossimità IR. Il sensore ad ultrasuoni consente al robot di evitare gli ostacoli in modo che possa muoversi liberamente fino a quando la stanza non sarà adeguatamente pulita, mentre il sensore di prossimità lo aiuterà ad evitare di cadere dalle scale. Questo dispositivo è una buona base di partenza per il nostro progetto, tuttavia non è in grado di mappare ambienti e di essere controllato in maniera remota, perciò subirà delle modifiche.

*Materiali necessari per costruire il robot di Mr.X*



* Arduino Pro Mini - 1
* HC-SR04 Ultrasonic Module - 3
* L293D Motor Driver - 1
* 5Volt N20 Motors and Mounting Brackets - 2
* N20 Motor Wheels - 2
* Switch - 1
* LM7805 Voltage Regulator - 1
* 7.4V Lithium-Ion Battery - 1
* IR Module - 1
* Perfboard - 1
* Castor Wheel - 1
* MDF
* Generic Portable Vacuum Cleaner

*Schema elettrico del robot aspirapolvere basato su Arduino*

****

## 

## 

## Software open source

Il codice presente nella PathFinding repository[[3]](#bookmark=id.2bn6wsx) è scritto in c++ ed è un algoritmo di Path Finding basato sul Breadth-First Search (BFS)[[4]](#bookmark=id.qsh70q) che ci dà la distanza dal punto di partenza al punto di arrivo all’interno di un array a due dimensioni tenendo conto di ostacoli.

Questo programma è adattabile alle nostre esigenze dal momento che, grazie ad esso, è possibile far arrivare il dispositivo a dei punti della casa che non sono ancora stati puliti nel minor tempo possibile evitando gli ostacoli.

Il codice contiene le seguenti funzioni:

* bool isValid, serve per controllare se delle coordinate di una cella siano valide;
* void display, serve a visualizzare nel terminale la matrice;
* int BFS, serve ad individuare la strada più corta tra una cella di partenza e una di arrivo.

Questo programma ci darà come output la distanza della strada più corta tra una cella di partenza e una di arrivo, ma può essere riadattato per individuare le singole celle che compongono la strada ignorando la sua lunghezza, il che sarebbe più utile per le nostre esigenze.

Alla base della funzione int BFS c’è la struttura dati queue la quale è di tipo first-in first-out (FIFO).

La queue è un contenitore nel quale possiamo inserire dati da un lato e estrarli dall’altro.

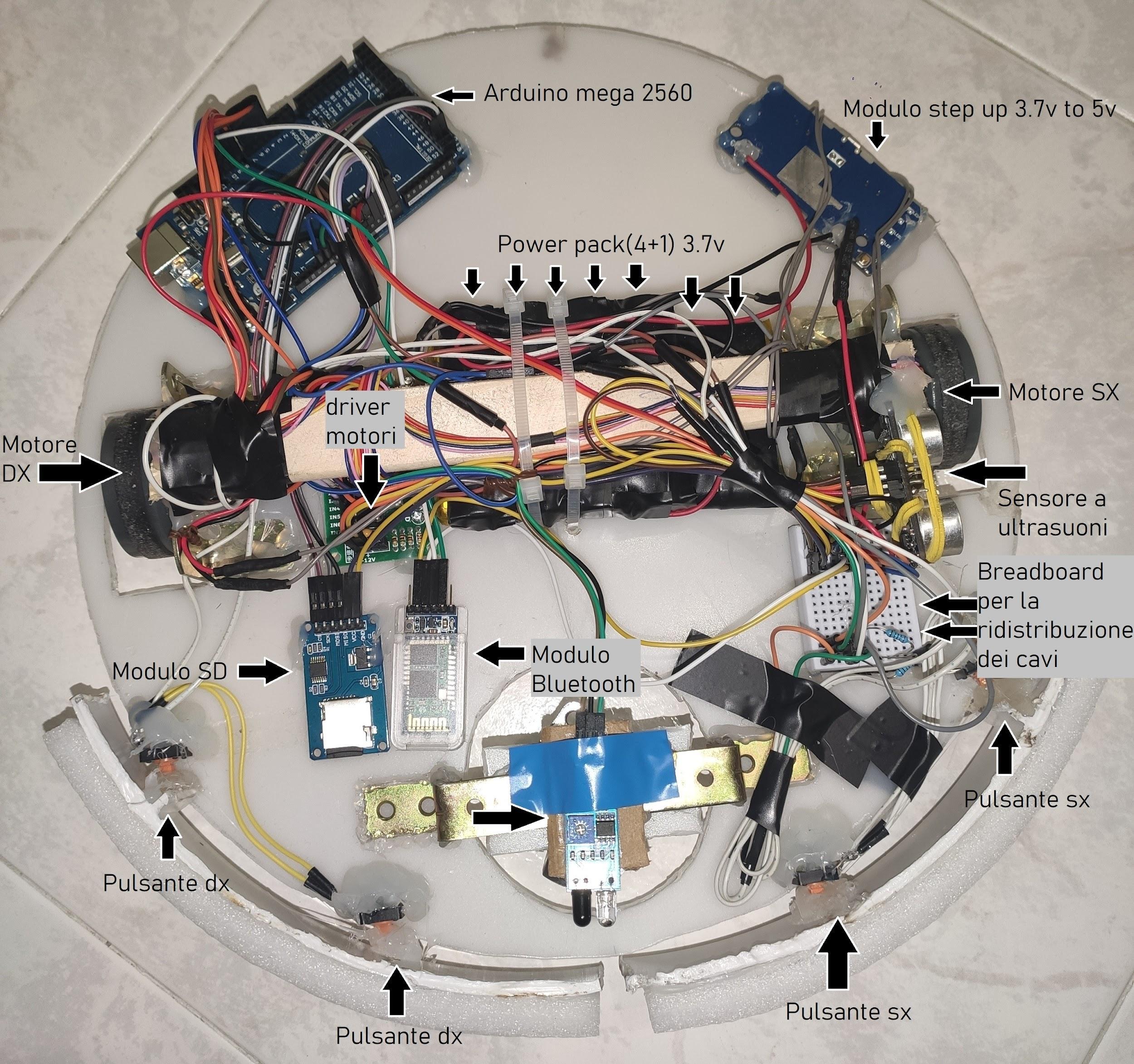
# **Disegno realizzazione e test**

## Specifica dei requisiti

Il robot aspirapolvere che vorremmo costruire sarà dotato di una funzione che consentirà la mappatura del luogo in cui si trova e la relativa memorizzazione in una scheda SD, poiché siamo interessati allo studio di questo tipo di algoritmi. Per fare questo sarà necessario insegnare al robot come girare su sé stesso di 90° in entrambi i versi, seguire un perimetro ed evitare gli ostacoli. Tuttavia ci sono funzioni secondarie che vorremmo inserire nell’elaborato finale quali:

1. aspirare la polvere;
2. controllo remoto via bluetooth;
3. memorizzare mappe relative a diverse strutture sullo stesso dispositivo;
4. rilevare automaticamente ostacoli ‘nuovi’ e apportare modifiche alla mappa.

## Progetto fisico

****

## Modello

* Modulo scheda madre:

La scheda utilizzata per questo progetto è la ELEGOO MEGA 2560 R3[[5]](#bookmark=id.89s9ek5xhjvp).

* Modulo motori:

Come motori si utilizzeranno due stepper motor 28BYJ-48[[6]](#bookmark=id.3as4poj), programmabili senza la necessità di installare particolari librerie. Come driver board useremo le ULN2003[[7]](#bookmark=id.1pxezwc).

* Modulo sensore ad ultrasuoni:

Il sensore ad ultrasuoni utilizzato è l’HC-SR04[[8]](#bookmark=id.49x2ik5), programmabile attraverso la libreria afstandssensor.

* Modulo alimentazione:

Per il progetto è stato utilizzato un “power pack” composto da una batteria 3.7V, 10.000mAh, 37.4Wh e 4 batterie 3.7V, 2700mAh, 10Wh poste in parallelo per sfruttare equamente tutte le batterie e fornire la quantità di ampere necessaria.

* Modulo step-up:

Il modulo step-up è sostanzialmente una scheda in grado di portare la tensione di 3.7V a 5V. Nello specifico è stata usata una scheda JX-887Y[[9]](#bookmark=id.e0fnkev2x2np) in grado di erogare 5V1A / 5V2A nelle sue porte di output. Questo modulo molto importante permette di alimentare dallo stesso pacco batterie (power pack) sia Arduino con i relativi sensori, sia le control board dei motori che richiedono una fonte “separata” e dedicata.

* Modulo bluetooth:

Il modulo usato è l’HC-05[[10]](#bookmark=id.48p8kfuqdntr) basato su Bluetooth SPP (Serial Port Protocol) che lavora su una frequenza di 2.4GHz e può trasmettere fino a 3Mbps.

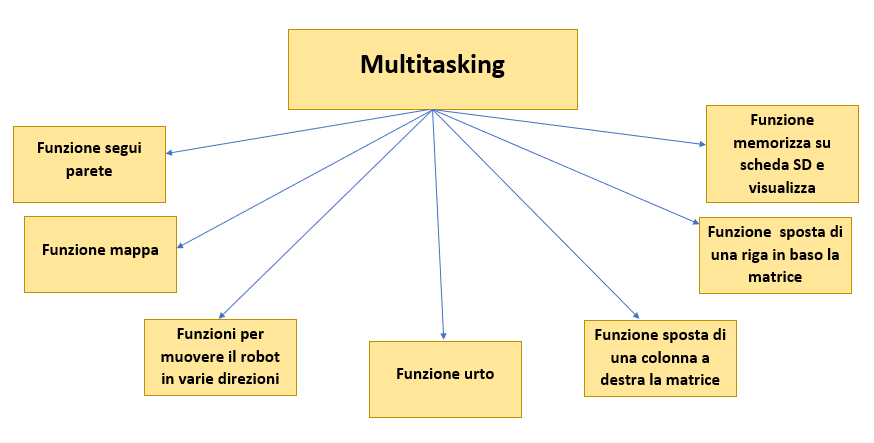
* Modulo scheda SD:

Il modulo per la lettura della scheda SD segue il tipo di formattazione fat16[[11]](#bookmark=id.qeyouekp78e4) che consente la lettura fino a 2GB.

* Modulo sensore infrarossi:

Il sensore ad infrarossi per rilevazione di ostacoli utilizzato è l’FC-51[[12]](#bookmark=id.gmp6zi14iy6j).

## Schema del software

****

Multitasking:

Con multitasking[[13]](#bookmark=id.v97k656h0eb2) (in italiano multiprocessualità) in informatica si indica la capacità di un software di eseguire più programmi contemporaneamente: se ad esempio viene chiesto al sistema di eseguire contemporaneamente due processi A e B, la CPU eseguirà per qualche istante di tempo il processo A, poi per qualche istante successivo il processo B, per poi tornare a eseguire il processo A e così via. Il passaggio dal processo A al processo B e viceversa viene definito "commutazione di contesto" (context switch). Le decisioni riguardanti l'esecuzione di un cambio di contesto tra due programmi vengono intraprese da un componente software del sistema operativo, lo scheduler, il quale invierà le proprie decisioni date principalmente dal nice a un altro modulo del sistema operativo, il dispatcher che eseguirà effettivamente il cambio di contesto: a seconda di quale strategia di servizio (algoritmo di scheduling) venga seguita, lo scheduler controlla la ripartizione del tempo di CPU tra tutti i processi attivi. Esistono due principali tecniche di controllo di termine e pausa del multitasking: il vecchio senza prelazione (cooperative) e il nuovo con prelazione (preemptive), ma oggigiorno la vera potenza del multitasking risiede solo nell'algoritmo di scheduling usato.

Funzione segui parete:

La funzione segui parete esegue istruzioni diverse a seconda della distanza rilevata dal sensore frontale e dal sensore infrarossi:

* se il sensore infrarossi rileva una distanza minore di 8 cm allora il robot effettua una svolta a destra;
* se la distanza laterale è minore di 7 cm allora il robot capirà di essere troppo vicino al muro e raddrizzerà la sua direzione;
* se la distanza laterale è maggiore di 8.5 cm e minore di 9.5 cm il robot capirà di essersi allontanato dal muro e raddrizzerà la sua direzione;
* se la distanza laterale è maggiore 9,5 cm e minore di 16 cm il robot eseguirà una serie di istruzioni che hanno lo scopo di avvicinarlo un po’ alla volta alla parete;
* se la distanza laterale è maggiore di 16 cm il robot effettua una svolta a sinistra.

Funzione mappa:

La mappatura è realizzata mediante l’uso di una matrice di char 70X70 (4.9KB) dove ogni cella corrisponde a 20 cm. Come grandezza massima sarà possibile mappare una struttura di 196 m2. La scheda SD utilizzata possiede una memoria di 2 GB sulla quale si possono memorizzare 40816 mappe. Il nostro robot può mappare in 4 modalità, una per ogni direzione. Di default il robot parte in modalità 0 ed ogni volta che effettua una svolta a destra, la modalità si incrementa di uno. L’incremento della modalità 3 imposta la modalità 0. Lo stesso principio è stato applicato alla svolta a sinistra ma anziché incrementare la modalità la si decrementa.

0 - In modalità zero il robot ogni 9 secondi registra il carattere W (wall) sulla matrice, incrementando di volta in volta solo l’indice delle colonne.

1 - In modalità uno il robot ogni 9 secondi registra il carattere W (wall) sulla matrice, incrementando di volta in volta solo l’indice delle righe.

2 - In modalità due il robot ogni 9 secondi registra il carattere W (wall) sulla matrice, decrementando di volta in volta solo l’indice delle colonne.

3 - In modalità tre il robot ogni 9 secondi registra il carattere W (wall) sulla matrice, decrementando di volta in volta solo l’indice delle righe.

Funzioni per muovere il robot in varie direzioni:

Attraverso una serie di istruzioni è possibile far girare di 90 gradi il robot a destra e a sinistra e mantenere un'andatura dritta.

Funzione urto:

Ogni qual volta i pulsanti di destra e sinistra vengono premuti viene effettuata una svolta a destra per consentire al robot di mappare l’intero perimetro evitando ostacoli posti su di esso.

Funzione sposta di una colonna a destra la matrice:

Se il robot mappa in modalità 2, allora può esserci un caso in cui l’indice delle colonne decrementando assume valore -1, in tal caso viene impostato a 0 e tutto ciò che è contenuto nella matrice viene spostato di una colonna a destra.

Funzione sposta di una riga in basso la matrice:

Se il robot mappa in modalità 3, allora può esserci un caso in cui l’indice delle righe decrementando assume valore -1, in tal caso viene impostato a 0 e tutto ciò che è contenuto nella matrice viene spostato di una riga in basso.

Funzione memorizza su scheda SD e visualizza:

Attraverso il modulo bluetooth ed un'applicazione di terze parti[[14]](#bookmark=id.ooy0koml43xj) è possibile, digitando 0, salvare l’intera matrice su scheda SD oppure, digitando 1, salvare la matrice e visualizzarla.

## Implementazioni

**Libreria multitasking AceRoutine**[**[15]**](#bookmark=id.2p2csry)**:**

Per utilizzare questa libreria è necessario un solo file di intestazione (AceRoutine.h). Se si dispone di un numero elevato di coroutine sarebbe meglio utilizzare: CoroutineScheduler::setup() che crea un elenco interno di coroutine attive gestite dallo scheduler e CoroutineScheduler::loop() che esegue una coroutine in quell'elenco con un algoritmo di pianificazione round-robin.

Esempio:

void setup() {

...

CoroutineScheduler::setup();

}

void loop() {

CoroutineScheduler::loop();

}

Macro:

COROUTINE(name): Istanza di classe Coroutine personalizzabile.

COROUTINE\_BEGIN(): Indica l'inizio della Coroutine.

COROUTINE\_END(): Indica la fine della Coroutine.

COROUTINE\_AWAIT(condition): Ignora il resto delle istruzioni se la condizione è falsa.

COROUTINE\_YIELD(): Restituisce il controllo a CoroutineScheduler che è quindi in grado di eseguire altre coroutine.

COROUTINE\_DELAY(millis): Ritarda l’esecuzione per tot millisecondi.

COROUTINE\_DELAY\_MICROS(micros): Ritarda l’esecuzione per tot microsecondi.

COROUTINE\_DELAY\_SECONDS(seconds): Ritarda l’esecuzione per tot secondi.

COROUTINE\_LOOP(): Esegue in loop le istruzioni.

//Coroutine that allows to follow a perimeter and manage the controller

**COROUTINE(follow\_wall) {**

COROUTINE\_LOOP() {

COROUTINE\_AWAIT(bwall); //Yields until condition become true

COROUTINE\_DELAY(1000);

isObstacle = digitalRead(isObstaclePin);

distance = afstandssensor2.afstandCM(); //Lateral distance from wall in cm

if (isObstacle == LOW) {

bright = true;

} else {

if (distance > 7 && distance < 8.5) {

bforward = true;

bcontroller = true;

} else {

if (distance < 7) {

bforwardr = true;

bcontroller = true;

} else {

if (distance > 8.5 && distance < 9.5) {

bforwardl = true;

bcontroller = true;

} else {

if (distance > 9.5 && distance < 16) {

bcontroller = false;

for (int i = 0; i < 680; i++) {

digitalWrite(m11, HIGH);

digitalWrite(m21, LOW);

digitalWrite(m12, LOW);

digitalWrite(m22, LOW);

digitalWrite(m13, LOW);

digitalWrite(m23, LOW);

digitalWrite(m14, LOW);

digitalWrite(m24, HIGH);

delay(motordelay);

digitalWrite(m11, LOW);

digitalWrite(m21, LOW);

digitalWrite(m12, HIGH);

digitalWrite(m22, LOW);

digitalWrite(m13, LOW);

digitalWrite(m23, HIGH);

digitalWrite(m14, LOW);

digitalWrite(m24, LOW);

delay(motordelay);

digitalWrite(m11, LOW);

digitalWrite(m21, LOW);

digitalWrite(m12, LOW);

digitalWrite(m22, HIGH);

digitalWrite(m13, HIGH);

digitalWrite(m23, LOW);

digitalWrite(m14, LOW);

digitalWrite(m24, LOW);

delay(motordelay);

digitalWrite(m11, LOW);

digitalWrite(m21, HIGH);

digitalWrite(m12, LOW);

digitalWrite(m22, LOW);

digitalWrite(m13, LOW);

digitalWrite(m23, LOW);

digitalWrite(m14, HIGH);

digitalWrite(m24, LOW);

delay(motordelay);

}

bforward = true;

COROUTINE\_DELAY(1000);

for (int i = 0; i < 620; i++) {

digitalWrite(m21, HIGH);

digitalWrite(m11, LOW);

digitalWrite(m22, LOW);

digitalWrite(m12, LOW);

digitalWrite(m23, LOW);

digitalWrite(m13, LOW);

digitalWrite(m24, LOW);

digitalWrite(m14, HIGH);

delay(motordelay);

digitalWrite(m21, LOW);

digitalWrite(m11, LOW);

digitalWrite(m22, HIGH);

digitalWrite(m12, LOW);

digitalWrite(m23, LOW);

digitalWrite(m13, HIGH);

digitalWrite(m24, LOW);

digitalWrite(m14, LOW);

delay(motordelay);

digitalWrite(m21, LOW);

digitalWrite(m11, LOW);

digitalWrite(m22, LOW);

digitalWrite(m12, HIGH);

digitalWrite(m23, HIGH);

digitalWrite(m13, LOW);

digitalWrite(m24, LOW);

digitalWrite(m14, LOW);

delay(motordelay);

digitalWrite(m21, LOW);

digitalWrite(m11, HIGH);

digitalWrite(m22, LOW);

digitalWrite(m12, LOW);

digitalWrite(m23, LOW);

digitalWrite(m13, LOW);

digitalWrite(m24, HIGH);

digitalWrite(m14, LOW);

delay(motordelay);

}

bcontroller = true;

} else {

if (distance > 16 && counter < 2) {

counter++;

if (counter == 2) {

bcontroller = false;

}

bforward = true;

COROUTINE\_DELAY(10000);

bleft = true;

}

}

}

}

}

}

COROUTINE\_YIELD(); //Yields execution back to the caller

}

}

//Coroutine that every 20 cm records the characters on the map depending on the mode set.

**COROUTINE(controller) {**

COROUTINE\_LOOP() {

COROUTINE\_AWAIT(bcontroller);

COROUTINE\_DELAY(9000);

switch (z) {

case 0:

y++;

matrix[x][y] = 'W';

break;

case 1:

x++;

matrix[x][y] = 'W';

break;

case 2:

y--;

if (y == -1) {

y = 0;

bcopyy = true;

} else {

matrix[x][y] = 'W';

}

break;

case 3:

x--;

if (x == -1) {

x = 0;

bcopyx = true;

} else {

matrix[x][y] = 'W';

}

break;

default:

break;

}

COROUTINE\_YIELD();

}

}

//Coroutine moving all matrix down one row

**COROUTINE(copy\_back\_x) {**

COROUTINE\_LOOP() {

COROUTINE\_AWAIT(bcopyx);

for (int i = 69; i >= 0; i--) {

for (int h = 0; h < 70; h++) {

if (i != 0) {

matrix[i][h] = matrix[i - 1][h];

} else {

matrix[i][h] = ' ';

}

}

}

matrix[x][y] = 'W';

bcopyx = false;

COROUTINE\_YIELD();

}

}

//Coroutine moving the entire matrix of a column to the right

**COROUTINE(copy\_back\_y) {**

COROUTINE\_LOOP() {

COROUTINE\_AWAIT(bcopyy);

for (int i = 0; i < 70; i++) {

for (int h = 69; h >= 0; h--) {

if (h != 0) {

matrix[i][h] = matrix[i][h - 1];

} else {

matrix[i][h] = ' ';

}

}

}

matrix[x][y] = 'W';

bcopyy = false;

COROUTINE\_YIELD();

}

}

**COROUTINE(forward) {**

COROUTINE\_LOOP() {

COROUTINE\_AWAIT(bforward);

bforwardr = false;

bforwardl = false;

//Code to turn the wheels and make the device go forward

//----------------------------------

digitalWrite(m11, LOW);

digitalWrite(m21, LOW);

digitalWrite(m12, LOW);

digitalWrite(m22, LOW);

digitalWrite(m13, LOW);

digitalWrite(m23, LOW);

digitalWrite(m14, HIGH);

digitalWrite(m24, HIGH);

delay(motordelay);

digitalWrite(m11, LOW);

digitalWrite(m21, LOW);

digitalWrite(m12, LOW);

digitalWrite(m22, LOW);

digitalWrite(m13, HIGH);

digitalWrite(m23, HIGH);

digitalWrite(m14, LOW);

digitalWrite(m24, LOW);

delay(motordelay);

digitalWrite(m11, LOW);

digitalWrite(m21, LOW);

digitalWrite(m12, HIGH);

digitalWrite(m22, HIGH);

digitalWrite(m13, LOW);

digitalWrite(m23, LOW);

digitalWrite(m14, LOW);

digitalWrite(m24, LOW);

delay(motordelay);

digitalWrite(m11, HIGH);

digitalWrite(m21, HIGH);

digitalWrite(m12, LOW);

digitalWrite(m22, LOW);

digitalWrite(m13, LOW);

digitalWrite(m23, LOW);

digitalWrite(m14, LOW);

digitalWrite(m24, LOW);

delay(motordelay);

COROUTINE\_YIELD();

//----------------------------------

}

}

//Coroutines that allows to correct the direction of the robot when it is away from the wall

COROUTINE(forward\_left) {

COROUTINE\_LOOP() {

COROUTINE\_AWAIT(bforwardl);

bforward = false;

bforwardr = false;

digitalWrite(m21, LOW);

digitalWrite(m22, LOW);

digitalWrite(m23, LOW);

digitalWrite(m24, HIGH);

delay(motordelay);

digitalWrite(m21, LOW);

digitalWrite(m22, LOW);

digitalWrite(m23, HIGH);

digitalWrite(m24, LOW);

delay(motordelay);

digitalWrite(m21, LOW);

digitalWrite(m22, HIGH);

digitalWrite(m23, LOW);

digitalWrite(m24, LOW);

delay(motordelay);

digitalWrite(m21, HIGH);

digitalWrite(m22, LOW);

digitalWrite(m23, LOW);

digitalWrite(m24, LOW);

delay(motordelay);

COROUTINE\_YIELD();

}

}

//Coroutines that allows to correct the direction of the robot when it is too close to the wall

**COROUTINE(forward\_right) {**

COROUTINE\_LOOP() {

COROUTINE\_AWAIT(bforwardr);

bforward = false;

bforwardl = false;

digitalWrite(m11, LOW);

digitalWrite(m12, LOW);

digitalWrite(m13, LOW);

digitalWrite(m14, HIGH);

delay(motordelay);

digitalWrite(m11, LOW);

digitalWrite(m12, LOW);

digitalWrite(m13, HIGH);

digitalWrite(m14, LOW);

delay(motordelay);

digitalWrite(m11, LOW);

digitalWrite(m12, HIGH);

digitalWrite(m13, LOW);

digitalWrite(m14, LOW);

delay(motordelay);

digitalWrite(m11, HIGH);

digitalWrite(m12, LOW);

digitalWrite(m13, LOW);

digitalWrite(m14, LOW);

delay(motordelay);

COROUTINE\_YIELD();

}

}

**COROUTINE(right) {**

COROUTINE\_LOOP() {

COROUTINE\_AWAIT(bright);

bcontroller = false;

bforwardr = false;

bforwardl = false;

bwall = false;

bshock = false;

bforward = false;

for (int i = 0; i < 680; i++) {

//Code to turn the wheels and make the device go right

//----------------------------------

digitalWrite(m21, HIGH);

digitalWrite(m11, LOW);

digitalWrite(m22, LOW);

digitalWrite(m12, LOW);

digitalWrite(m23, LOW);

digitalWrite(m13, LOW);

digitalWrite(m24, LOW);

digitalWrite(m14, HIGH);

delay(motordelay);

digitalWrite(m21, LOW);

digitalWrite(m11, LOW);

digitalWrite(m22, HIGH);

digitalWrite(m12, LOW);

digitalWrite(m23, LOW);

digitalWrite(m13, HIGH);

digitalWrite(m24, LOW);

digitalWrite(m14, LOW);

delay(motordelay);

digitalWrite(m21, LOW);

digitalWrite(m11, LOW);

digitalWrite(m22, LOW);

digitalWrite(m12, HIGH);

digitalWrite(m23, HIGH);

digitalWrite(m13, LOW);

digitalWrite(m24, LOW);

digitalWrite(m14, LOW);

delay(motordelay);

digitalWrite(m21, LOW);

digitalWrite(m11, HIGH);

digitalWrite(m22, LOW);

digitalWrite(m12, LOW);

digitalWrite(m23, LOW);

digitalWrite(m13, LOW);

digitalWrite(m24, HIGH);

digitalWrite(m14, LOW);

delay(motordelay);

//----------------------------------

}

if (counter != 2) {

bcontroller = true;

z++;

if (z == 4) {

z = 0;

}

} else {

counter = 0;

}

bright = false;

bwall = true;

bshock = true;

COROUTINE\_YIELD();

}

}

**COROUTINE(left) {**

COROUTINE\_LOOP() {

COROUTINE\_AWAIT(bleft);

bforward = false;

bcontroller = false;

bwall = false;

bshock = false;

bforwardr = false;

bforwardl = false;

for (int i = 0; i < 680; i++) {

//Code to turn the wheels and make the device go left

//----------------------------------

digitalWrite(m11, HIGH);

digitalWrite(m21, LOW);

digitalWrite(m12, LOW);

digitalWrite(m22, LOW);

digitalWrite(m13, LOW);

digitalWrite(m23, LOW);

digitalWrite(m14, LOW);

digitalWrite(m24, HIGH);

delay(motordelay);

digitalWrite(m11, LOW);

digitalWrite(m21, LOW);

digitalWrite(m12, HIGH);

digitalWrite(m22, LOW);

digitalWrite(m13, LOW);

digitalWrite(m23, HIGH);

digitalWrite(m14, LOW);

digitalWrite(m24, LOW);

delay(motordelay);

digitalWrite(m11, LOW);

digitalWrite(m21, LOW);

digitalWrite(m12, LOW);

digitalWrite(m22, HIGH);

digitalWrite(m13, HIGH);

digitalWrite(m23, LOW);

digitalWrite(m14, LOW);

digitalWrite(m24, LOW);

delay(motordelay);

digitalWrite(m11, LOW);

digitalWrite(m21, HIGH);

digitalWrite(m12, LOW);

digitalWrite(m22, LOW);

digitalWrite(m13, LOW);

digitalWrite(m23, LOW);

digitalWrite(m14, HIGH);

digitalWrite(m24, LOW);

delay(motordelay);

//----------------------------------

}

if (counter == 1) {

z--;

if (z == -1) {

z = 3;

}

bcontroller = true;

}

bleft = false;

bforward = true;

bwall = true;

bshock = true;

COROUTINE\_YIELD();

}

}

//Coroutine that allows you to save the map on sd card or to save the map and to show it on the terminal

**COROUTINE(Sd\_bluetooth) {**

COROUTINE\_LOOP() {

COROUTINE\_AWAIT(bsd);

if (BTserial.available()) { //wait for data received

Data = BTserial.read();

if (Data == '0') {

if (counter2 != 0) {

SD.remove("Mappa.txt");

}

counter2 = 1;

myMap = SD.open("Mappa.txt", FILE\_WRITE);

if (myMap) {

for (int h = 0; h < 70; h++) {

for (int i = 0; i < 70; i++) {

myMap.print(matrix[h][i]);

}

myMap.println(" ");

}

}

myMap.close();

BTserial.println("Map completed");

} else {

if (Data == '1') {

if (counter2 != 0) {

SD.remove("Mappa.txt");

}

counter2 = 1;

myMap = SD.open("Mappa.txt", FILE\_WRITE);

if (myMap) {

for (int h = 0; h < 70; h++) {

for (int i = 0; i < 70; i++) {

myMap.print(matrix[h][i]);

}

myMap.println(" ");

}

myMap.close();

}

myMap = SD.open("Mappa.txt", FILE\_READ);

if (myMap) {

while (myMap.available()) {

convert = myMap.read();

BTserial.print(convert);

}

myMap.close();

}

}

}

}

COROUTINE\_YIELD();

}

}

//Coroutine that allows to avoid obstacles along the perimeter

**COROUTINE(shock) {**

COROUTINE\_LOOP() {

COROUTINE\_AWAIT(bshock);

COROUTINE\_DELAY(200);

//Reading the status of the buttons on the left

int fbs = digitalRead(fbutton);

//Reading the status of the buttons on the right

int sbs = digitalRead(sbutton);

if (fbs == 1 || sbs == 1) {

bright = true;

}

COROUTINE\_YIELD();

}

}

## Test

Test AceRoutine.h: [VID\_20221112\_202040.mp4](https://drive.google.com/file/d/1blxW5rVmyR8f358wUC3tZBL22j_ZWWv5/view?usp=share_link)

Test funzione segui parete: [VID\_20230124\_200230.mp4](https://drive.google.com/file/d/1_C0_3pZegxNIx73P-gzoRMxtoE0pqMYg/view?usp=share_link)

Test funzione mappa e memorizza su scheda SD e visualizza: [Screenrecorder-2023-01-23-16-30-12-297.mp4](https://drive.google.com/file/d/1Zq4woHXcGJfq3b6swq_Lpxrh2ZslDiZ0/view?usp=share_link)

Test funzione sposta la matrice di una colonna/riga a destra/in basso la matrice: [spostamatrice.c](https://drive.google.com/file/d/15Drghom0qISeDXSMyZ8F5luQPhLJDJRL/view?usp=share_link)

Test funzione urto: [VID\_20230123\_162707.mp4](https://drive.google.com/file/d/1ZrpywQLOtfMzK94wzeQ-h5ITtKS-u62o/view?usp=share_link)

# **Conclusione e sviluppi futuri**

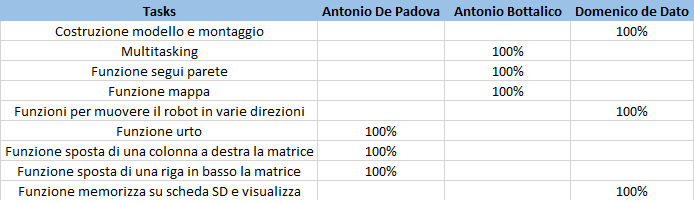
Il nostro robot è in grado di percorrere un perimetro evitando ostacoli posti su di esso, registrarlo in una matrice e farlo visualizzare all’utente.

Miglioramenti e ulteriori implementazioni:

* sostituzione della struttura, con una stampata in 3D o riciclata da un robot rotto;
* utilizzo di un accelerometro per essere sicuri di aver percorso 20 cm;
* reimpiego dei sensori in maniera tale da poter distinguere l’impatto con un ostacolo o con la parete;
* cambio di motori con un paio più potenti;
* implementazione dell’algoritmo che indica il miglior percorso da seguire all’interno della matrice, rilevando gli ostacoli;
* implementazione dell’algoritmo che permette di generare ogni 10 accensioni un differente file su cui memorizzare la mappa.

In conclusione ci riteniamo soddisfatti dell’andamento del nostro progetto e della mole di nozioni imparate, sempre aperti ad acquisire ulteriori conoscenze.

# **Matrice RACI**



# **Bibliografia e sitografia**

1. Chi ha inventato il robot aspirapolvere?,

<<https://www.robotaspirapolvere.info/chi-ha-inventato-il-robot-aspirapolvere/>>.

1. Build your own Arduino based Smart Vacuum Cleaner Robot for Automatic Floor Cleaning,

<<https://circuitdigest.com/microcontroller-projects/build-your-own-arduino-based-smart-vacuum-cleaning-robot-for-automatic-floor-cleaning>>.

1. PathFinding,

<<https://github.com/sakky016/PathFinding>>.

1. Ricerca in ampiezza (Breadth-first search),

<<https://it.wikipedia.org/wiki/Ricerca_in_ampiezza>>.

1. Elegoo Mega 2560 R3 Data sheet,

<<https://www.arsbook.it/stempdf/LS-ELEGOO-MEGA.pdf>>

1. Stepper Motor Data Sheet, 28BYJ-48,

<<https://www.mouser.com/datasheet/2/758/stepd-01-data-sheet-1143075.pdf>>

1. Driver Board Data Sheet, ULN2003,

<<https://www.hadex.cz/spec/m513.pdf>>.

1. Sensore ad Ultrasuoni Data Sheet, HC-SR04,

<<https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>>.

1. Step up Data Sheet, JX-887Y,

<<https://electronicspices.com/product/jx-887y-dual-micro-usb-3-7v-to-5v-2a-power-bank-chargers-modules-pack-of-50pcs>>.

1. Modulo Bluetooth Data Sheet, HC-05,

<<https://components101.com/sites/default/files/component_datasheet/HC-05%20Datasheet.pdf>>.

1. FAT16,

<<https://it.ccm.net/contents/251-fat16-e-fat32#il-file-system-fat16>>.

1. Modulo sensore infrarossi, FC-51,

<<http://www.dmf.unisalento.it/~denunzio/allow_listing/ARDUINO/FC51.pdf>>

1. Multitasking,

<<https://it.wikipedia.org/wiki/Multitasking#:~:text=Con%20multitasking%20(in%20italiano%20multiprocessualit%C3%A0,processo%20A%2C%20poi%20per%20qualche>>.

1. Serial Bluetooth Terminal,

<<https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal&hl=it&gl=US>>.

1. AceRoutine User Guide,

<<https://github.com/bxparks/AceRoutine/blob/develop/USER_GUIDE.md>>.

# **Appendici**

**A - Relazioni/Documenti di progetto**

Relazione del 12/10/2022: [Rel. 12-10-22.docx](https://docs.google.com/document/d/1sy0b2QyqgFVlKGelewtgw70KbWdMPGpQ/edit?usp=share_link&ouid=104817336570539172518&rtpof=true&sd=true)

D.D.P versione 1: [Versione 1](https://docs.google.com/document/d/1ATOnq-kRQuEgy1pzbFJQIQocHc8-UtlE/edit?usp=share_link&ouid=104817336570539172518&rtpof=true&sd=true)

D.D.P versione 2: [Versione 2](https://docs.google.com/document/d/1WrOhkXSiV132nln7FCFRL9IxSsU-Lv_y/edit?usp=share_link&ouid=104817336570539172518&rtpof=true&sd=true)

D.D.P versione 3: [Versione 3](https://docs.google.com/document/d/13k-AUhn_UxVcTETt4OxKh0iCCNWLZdy1/edit?usp=share_link&ouid=104817336570539172518&rtpof=true&sd=true)

D.D.P versione 4: [Versione 4](https://docs.google.com/document/d/1fi8K9ddDXYesqBSWTqGBefAeVXU3dxTW/edit?usp=share_link&ouid=104817336570539172518&rtpof=true&sd=true)

**B - Minute**

Minuta del 11/10/2022: [Minuta del 11/10/2022.docx](https://docs.google.com/document/d/13s1D_N5LC4kKVv7cYsPxM35RQwFC88La/edit?usp=share_link&ouid=104817336570539172518&rtpof=true&sd=true)

Minuta del 12/10/2022: [Minuta del 12/10/2022.docx](https://docs.google.com/document/d/13F7ELUU9vjtBsnoY33TIP81S1oalvSP9/edit?usp=share_link&ouid=104817336570539172518&rtpof=true&sd=true)

Minuta del 20/10/2022: [Minuta del 20/10/2022.docx](https://docs.google.com/document/d/1UHe_FVgNcXf65O5ou9EcpCus-eKDza3K/edit?usp=share_link&ouid=104817336570539172518&rtpof=true&sd=true)

Minuta del 03/11/2022: [Minuta del 03/11/2022.docx](https://docs.google.com/document/d/1Cn-7OLZRYeXrOQ1HLoItJqO1Grsz6_OR/edit?usp=share_link&ouid=104817336570539172518&rtpof=true&sd=true)

Minuta del 17/11/2022: [Minuta del 17/11/2022.docx](https://docs.google.com/document/d/1ufuo-Fwqt4rXBrSXPVW0xvYUxIQozqq5/edit?usp=share_link&ouid=104817336570539172518&rtpof=true&sd=true)

Minuta del 24/11/2022: [Minuta del 24/11/2022.docx](https://docs.google.com/document/d/1kz0K3QOjSJRhiBNOOVIQdpNvahZ2qLp6/edit?usp=share_link&ouid=104817336570539172518&rtpof=true&sd=true)

Minuta del 01/12/2022: [Minuta del 01/12/2022.docx](https://docs.google.com/document/d/1icRRxAJ9p8-GayNhwOGazVWyZw6S3uZX/edit?usp=share_link&ouid=104817336570539172518&rtpof=true&sd=true)

**C - Librerie**

Libreria per il sensore ad ultrasuoni: [Afstandssensor\_-\_HCSR04.rar](https://drive.google.com/file/d/1KzDyQJIZzfyrOh68iBljxQOMoY7g1srk/view?usp=share_link)

Librerie per il multitasking: [AceCommon.rar](https://drive.google.com/file/d/14pnHBtWN0a3ZZS3fNzki-PLJo9_4MivT/view?usp=share_link) - [AceRoutine.rar](https://drive.google.com/file/d/15-GgQyZ-P0vrrKQiTEQ3pY0hNFQg3LM8/view?usp=share_link)

**D - Codici**

Main: [main4](https://drive.google.com/drive/folders/14MSZLg1HTVo492A63V0kZzYcecLjtfl-?usp=share_link)

Motori: [motori.ino](https://drive.google.com/drive/folders/14MSZLg1HTVo492A63V0kZzYcecLjtfl-?usp=share_link)

Sensori ultrasuoni: [sensori\_ultrasuoni.ino](https://drive.google.com/drive/folders/1gcEGgoOgdde0iFOCEM23G4dRMFoSfv_K?usp=share_link)

Scheda SD: [SD.ino](https://drive.google.com/drive/folders/1YGQvXYvL_Yj5qRGgqNYOc0wGneAkYILj?usp=share_link)

Scheda SD con bluetooth: [sd\_bluetooth.ino](https://drive.google.com/drive/folders/14HxhHuceiai_hetMXEDVLvr3fVr64S30?usp=share_link)